UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/723,967	11/26/2003	Joseph G. Laura	IDF 2584 (4000-16100)	9521

28003 7590 04/16/2008

SPRINT
6391 SPRINT PARKWAY
KSOPHT0101-Z2100
OVERLAND PARK, KS 66251-2100

EXAMINER	
WANG, BEN C	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
04/16/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/723,967	LAURA, JOSEPH G.	
	Examiner	Art Unit	
	BEN C. WANG	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *20 November 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-38* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-38* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *11/20/2007*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's amendment dated November 20, 2007, responding to the Office action mailed August 23, 2007 provided in the rejection of claims 1-38, wherein claims 21-23, 28, 30-32, and 35 have been amended.

Claims 1-38 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims currently amended have been fully considered but are moot in view of the new grounds of rejection – see *Jie Tao et al., (Tao-2)* art made of record, as applied hereto.

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-4, 6-11, 12-14, and 17-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sridharan et al. (*On Building Non-Intrusive Performance Instrumentation Blocks for CORBA-based Distributed Systems, March 2000, IEEE*) (hereinafter 'Sridharan') in view of Tao et al. (*Understanding the Behavior of Shared Memory Applications Using the SMiLE Monitoring Framework, March 2000, IEEE*) (hereinafter 'Tao-1')

3. **As to claim 1** (Previously Presented), Sridharan discloses a system for non-intrusively monitoring an application (e.g., Abstract – a non-intrusive, reusable framework for collecting performance statistics of CORBA-based distributed systems is proposed; Sec. 2 – Properties for Monitoring System), comprising:

- a first module stored on a computer-readable medium that attaches to a memory area that is used by an application during real-time operation, the first module reads application values from the memory area that have been stored in the memory area by the application during real-time operation (e.g., Fig. 3 – System T and Performance Instrumentation, element of "PI"; P. 3, 1st Par. – The Visibroker ORB,

which was used to deploy the servers, provided the facility to load the Performance Instrumentation (PI) module to together with the server into a single address space while staring the server using the Java VM; P. 3, 2nd Par. – 4th Par.);

- a second module stored on a computer-readable medium in communication with the first module that requests the first module to read the application values (e.g., Fig. 3 – System T and Performance Instrumentation, element of "ORB"; Sec. 5 – Instrumentation Framework, 1st Par. – The CORBA Interceptor facility allows interception of a CORBA call made to a server either at the client or server side; the interceptor facility is a CORBA implementation or ORB's feature); and
- a third module stored on a computer-readable medium in communication with the second module, that displays the application values (e.g., Fig. 3 – System T and Performance Instrumentation, element of "PM GUI"; P. 3, last Par. – A GUI was implemented to display the various performance data, to set the filtering information and turn on and off the monitoring mechanism).

Sridharan does not explicitly disclose the second module receives the application values from the first module.

However, in an analogous art of *Understanding the Behavior of Shared Memory Applications Using the SMiLE Monitoring Framework*, Tao-1 discloses the second module receives the application values from the first module (e.g., Sec. IV. – Designing A Tool Environment on Top of The SmiLE Monitoring Approach", 5th Par. – A mechanism for mapping each memory location observed by the monitoring system to its

corresponding program data structure identifier (procedure and variable names) is being implemented as well).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Tao-1 into the Sridharan's system to further provide the second module receives the application values from the first module in Sridharan system.

The motivation is that it would further enhance the Sridharan's system by taking, advancing and/or incorporating Tao-1's system which offers significant advantages that it provides the programmer or system software with detailed information about all memory transactions over the network and the behavior of user-defined events as once suggested by Tao-1 (e.g., Sec. I. – Motivation, 3rd Par., Lines 18-26).

4. **As to claim 2** (Original), Tao-1 discloses the system wherein the memory area is further defined as a shared memory of the application (e.g., Abstract).

5. **As to claim 3** (Original), Sridharan discloses the system wherein the first module is further operable to attach to the memory area used by the application to read the application values (e.g., Fig. 4 – Performance Instrumentation (PI) Module – White Box View, element "I"; P. 3, 4th Par.).

6. **As to claim 4** (Original), Tao-1 discloses the system wherein the application values are further defined as at least one application variable and a value for the

application variable (e.g., Sec. IV. – Designing A Tool Environment on Top of The SmiLE Monitoring Approach", 5th Par. – A mechanism for mapping each memory location observed by the monitoring system to its corresponding program data structure identifier (procedure and variable names) is being implemented as well).

7. **As to claim 6** (Original), Sridharan discloses the system wherein the third module is further defined as a graphical user interface (e.g., Sec. IV. – Designing A Tool Environment on Top of The SmiLE Monitoring Approach", 5th Par. – A mechanism for mapping each memory location observed by the monitoring system to its corresponding program data structure identifier (procedure and variable names) is being implemented as well).

8. **As to claim 7** (Original), Sridharan discloses the system wherein the graphical user interface is further operable to receive an input identifying the application values to be read and operable to request the application values identified to the first module (e.g., Fig. 3 – System T and Performance Instrumentation, element of "PI"; Fig. 4 – Performance Instrumentation (PI) Module – White Box View, element "I"; P. 3, 7th Par – In Fig. 4, the thread labeled T1 posts the reception of the request $P(P_1,P_2)$ as an event in the event service; P. 3, 1st Par. – The Visibroker ORB, which was used to deploy the servers, provided the facility to load the Performance Instrumentation (PI) module to together with the server into a single address space while staring the server using the Java VM; P. 3, 2nd Par. – 4th Par.), via the second module (e.g., Fig. 3 – System T and

Performance Instrumentation, element of "ORB"; Sec. 5 – Instrumentation Framework, 1st Par. – The CORBA Interceptor facility allows interception of a CORBA call made to a server either at the client or server side; the interceptor facility is a CORBA implementation or ORB's feature), and wherein the first module is operable to read the requested application values data from the memory area and return the application variables to the graphical user interface (e.g., Fig. 3 – System T and Performance Instrumentation, element of "PM GUI"; P. 3, last Par. – A GUI was implemented to display the various performance data, to set the filtering information and turn on and off the monitoring mechanism), via the second module.

9. **As to claim 8** (Original), Sridharan discloses the system wherein the graphical user interface is further operable to receive an input identifying requested application values to be displayed (e.g., Sec. 2, 4th Para. – cooperates with Graphical User Interface in the process of information visualization, implements the selective monitoring policy, and manages domains of local monitors; Sec. 5, 1st Para., 2nd Para. – GUI's main role is visualization of behavior of monitored systems' selected parts).

10. **As to claim 9** (Original), Sridharan discloses the system wherein the first module is further operable as a socket server and wherein the second module is further operable as a socket client such that the first and second modules communicate via a socket connection (e.g., Fig. 3 – System T and Performance Instrumentation, element of "PM GUI"; P. 3, last Par. – A GUI was implemented to display the various

performance data, to set the filtering information and turn on and off the monitoring mechanism).

11. **As to claim 10** (Original), Sridharan discloses the system wherein the first module operable to read application values stored in the memory area by the application while the application is running (e.g., Fig. 3 – System T and Performance Instrumentation, element of "PI"; P. 3, 1st Par. – The Visibroker ORB, which was used to deploy the servers, provided the facility to load the Performance Instrumentation (PI) module to together with the server into a single address space while staring the server using the Java VM; P. 3, 2nd Par. – 4th Par.).

12. **As to claim 11** (Original), Sridharan discloses the system wherein first module operable to read application values stored in the memory area by the application without interfering with the operation of the application (e.g., Sec. 2 – Properties for Monitoring System, bullet 1 – bullet 3 – non-intrusiveness; selective on-line monitoring; selective on-line filtering).

13. **As to claim 12** (Previously Presented), Sridharan discloses a method of non-intrusively monitoring operation of an application (e.g., Abstract – a non-intrusive, reusable framework for collecting performance statistics of CORBA-based distributed systems is proposed; Sec. 2 – Properties for Monitoring System), comprising:

- running an application in a real-time manner;

- generating, by the application, application values during operation of the application (e.g., Fig. 3 – System T and Performance Instrumentation, elements of "S1", "S2", and "S3");
- storing, by the application, the application values in a memory area during the operation of the application;
- displaying the application values read from the memory area (e.g., Fig. 3 – System T and Performance Instrumentation, element of "PM GUI"; P. 3, last Par. – A GUI was implemented to display the various performance data, to set the filtering information and turn on and off the monitoring mechanism).

Further, Sridharan does not explicitly disclose reading, by a monitor, the memory area used by the application to obtain the application values, wherein at least one of the application values is not output by the application;

However, in an analogous art of *Understanding the Behavior of Shared Memory Applications Using the SMiLE Monitoring Framework*, Tao-1 discloses reading, by a monitor, the memory area used by the application to obtain the application values, wherein at least one of the application values is not output by the application (e.g., Sec. IV. – Designing A Tool Environment on Top of The SmiLE Monitoring Approach", 5th Par. – A mechanism for mapping each memory location observed by the monitoring system to its corresponding program data structure identifier (procedure and variable names) is being implemented as well).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Tao-1 into the Sridharan's

system to further provide reading, by a monitor, the memory area used by the application to obtain the application values, wherein at least one of the application values is not output by the application in Sridharan system.

The motivation is that it would further enhance the Sridharan's system by taking, advancing and/or incorporating Tao-1's system which offers significant advantages that it provides the programmer or system software with detailed information about all memory transactions over the network and the behavior of user-defined events as once suggested by Tao-1 (e.g., Sec. I. – Motivation, 3rd Par., Lines 18-26).

14. **As to claim 13** (Previously Presented), Sridharan discloses the method further comprising: requesting, by a client, application values from the monitor (e.g., Fig. 3 – System T and Performance Instrumentation, element of "PI"; P. 3, 1st Par. – The Visibroker ORB, which was used to deploy the servers, provided the facility to load the Performance Instrumentation (PI) module to together with the server into a single address space while staring the server using the Java VM; P. 3, 2nd Par. – 4th Par.); and communicating the application variables from the monitor to the client (e.g., Fig. 3 – System T and Performance Instrumentation, element of "ORB"; Sec. 5 – Instrumentation Framework, 1st Par. – The CORBA Interceptor facility allows interception of a CORBA call made to a server either at the client or server side; the interceptor facility is a CORBA implementation or ORB's feature).

15. **As to claim 14** (Previously Presented), Sridharan discloses the method further comprising: requesting application values; running a plurality of applications in a real-time manner; generating application values stored in one or more memory areas during operation of the plurality of applications; reading the one or more memory areas used by the plurality of applications to obtain the application values (e.g., Fig. 3 – System T and Performance Instrumentation, element of "PI"; P. 3, 1st Par. – The Visibroker ORB, which was used to deploy the servers, provided the facility to load the Performance Instrumentation (PI) module to together with the server into a single address space while staring the server using the Java VM; P. 3, 2nd Par. – 4th Par.); and displaying the requested application values (e.g., Fig. 3 – System T and Performance Instrumentation, element of "ORB"; Sec. 5 – Instrumentation Framework, 1st Par. – The CORBA Interceptor facility allows interception of a CORBA call made to a server either at the client or server side; the interceptor facility is a CORBA implementation or ORB's feature).

16. **As to claim 17** (Original), Tao-1 discloses the method further comprising: generating new application values by the application stored in the memory area, at least one of the new application values defined as a new value for a variable of the application; requesting, by the client, that the monitor re-read the application values stored in the memory area; re-reading, by the monitor, the memory area to obtain the new application values (e.g., Sec. IV. – Designing A Tool Environment on Top of The SmiLE Monitoring Approach", 5th Par. – A mechanism for mapping each memory

location observed by the monitoring system to its corresponding program data structure identifier (procedure and variable names) is being implemented as well).

17. **As to claim 18** (Original), Tao discloses the method wherein the monitor reads the application values while the application is running ((e.g., Sec. IV. – Designing A Tool Environment on Top of The SmiLE Monitoring Approach", 5th Par. – A mechanism for mapping each memory location observed by the monitoring system to its corresponding program data structure identifier (procedure and variable names) is being implemented as well).

18. **As to claim 19** (Original), Sridharan discloses the method wherein the monitor is operable as a socket server and wherein the client is operable as a socket client such that the communication between the monitor and client is via a socket connection (e.g., Fig. 3 – System T and Performance Instrumentation, element of "ORB"; Sec. 5 – Instrumentation Framework, 1st Par. – The CORBA Interceptor facility allows interception of a CORBA call made to a server either at the client or server side; the interceptor facility is a CORBA implementation or ORB's feature).

19. **As to claim 20** (Original), Tao-1 discloses the method wherein the application values are further defined as a variable of the application and a value of the variable (e.g., Sec. IV. – Designing A Tool Environment on Top of The SmiLE Monitoring Approach", 5th Par. – A mechanism for mapping each memory location observed by the

monitoring system to its corresponding program data structure identifier (procedure and variable names) is being implemented as well)

20. Claims 5 and 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sridharan in view of Tao-1 and further in view of Hiroshi Kashima (*An Approach for Constructing Web Enterprise Systems on Distributed Objects, Jan., 2000, IBM*) (hereinafter 'Kashima')

21. **As to claim 5** (Original), Sridharan and Tao-1 do not disclose the system wherein the first module is further operable to communicate the application values to the second module in hypertext markup language format.

However, in an analogous art of *an approach for constructing web enterprise systems on distributed objects*, Kashima discloses the system wherein the first module is further operable to communicate the application values to the second module in hypertext markup language format (i.e., Abstract, 1st Para.; Sec. 1-2, 1st Para.; Sec. 1-3, 5th Para., Lines 1-3).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Sridharan-Tao-1 and the teachings of Kashima to further provide the system wherein the first module is further operable to communicate the application values to the second module in hypertext markup language format in Sridharan-Tao-1 system.

The motivation is that it would enhance the Sridharan-Tao-1 system by taking, advancing and/or incorporating the shared data which will enhance the expandability as a central repository, CORBA (common Object Request Broker Architecture) technologies, and COBOL support which is very important for the products that support mainframe systems as once suggested by Kashima (i.e., Abstract, 2nd Para.; Sec. 2, 2nd Para; Sec. 3-5, sub-sec. of 'shared data'; Sec. 2-1, sub-sec. of 'Language Support').

22. **As to claim 15** (Original), Sridharan and Tao-1 do not disclose the method wherein the memory area is further defined as a block of shared memory and wherein the monitor reads the at least some of the application variables stored in the block of shared memory.

However, in an analogous art of *an approach for constructing web enterprise systems on distributed objects*, Kashima discloses the method wherein the memory area is further defined as a block of shared memory and wherein the monitor reads the at least some of the application variables stored in the block of shared memory (e.g., Sec. 3-5, sub-sec. of Shared Data – this will enhance the expandability if it is centrally provided as in the concept of a repository, because any number of applications can access it).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Sridharan-Tao-1 and the teachings of Kashima to further provide the method wherein the memory area is further defined as a block of shared memory and wherein the monitor reads the at least some

of the application variables stored in the block of shared memory in Sridharan-Tao-1 system.

The motivation is that it would enhance the Sridharan-Tao-1 system by taking, advancing and/or incorporating the shared data which will enhance the expandability as a central repository, CORBA (common Object Request Broker Architecture) technologies, and COBOL support which is very important for the products that support mainframe systems as once suggested by Kashima (i.e., Abstract, 2nd Para.; Sec. 2, 2nd Para; Sec. 3-5, sub-sec. of 'shared data'; Sec. 2-1, sub-sec. of 'Language Support').

23. Claim 16 is rejected under 35 U.S.C. 103(a) as being unpatentable over Sridharan in view of Tao-1 and further in view of Huang et al., (*Operating System Support for Flexible Coherence in Distributed Shared Memory, 1996, IEEE*) (hereinafter 'Huang')

24. **As to claim 16** (Original), Sridharan does not disclose the method further comprising providing memory manager and wherein the monitor registers with the memory manager to obtain a location of the memory area used by the application to store the application values.

However, in an analogous art of *Operating System Support for Flexible Coherence in Distributed Shared Memory*, Huang discloses the method further comprising providing memory manager and wherein the monitor registers with the memory manager to obtain a location of the memory area used by the application to

store the application values (e.g., Fig. 2 – COMMOS Architecture, element of "Object Manager"; Sec. 3.4 – The COMMOS Architecture, 3rd Par.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Huang into the Sridharan's system to further provide the method further comprising the step of using an adaptation mode selected from the group comprising: on demand; and autonomous in Sridharan system.

The motivation is that it would further enhance the Sridharan's system by taking, advancing and/or incorporating Huang's system which offers significant advantages that the major advantages are the openness and ability to enforce modularity behind memory protection boundaries, sometimes characterized as the separation of policy from mechanism as once suggested by Huang (e.g., Sec. 3.1 - Microkernels).

25. Claims 21, 23, 26, 28, and 32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sridharan in view of Jie Tao et al. *(Visualizing the Memory Access Behavior of Shared Memory Applications on NUMA Architectures, Springer-Verlag Berlin Heidelberg 2001, pp. 861-870)* (hereinafter 'Tao-2' - art made of record)

26. **As to claim 21** (Currently Amended), Sridharan discloses a system for non-intrusively monitoring variables during operation of an application (e.g., Abstract – a non-intrusive, reusable framework for collecting performance statistics of CORBA-based distributed systems is proposed; Sec. 2 – Properties for Monitoring System).

Sridharan does not explicitly disclose a compile listing stored on a computer-readable medium having an address map with an offset associated with each of a plurality of variable of an application; and a module stored on a computer-readable medium that performs reading of the compile listing and obtaining the offset of at least one of the plurality of variables of the application, the module performs attaching to an address space used by the application during real-time operation to obtain a value for one or more the plurality of variables during the real-time operation of the application using the offset.

However, in an analogous art of *Visualizing the Memory Access Behavior of Shared Memory Applications on NUMA Architectures*, Tao-2 discloses a compile listing stored on a computer-readable medium having an address map with an offset associated with each of a plurality of variable of an application; and a module stored on a computer-readable medium that performs reading of the compile listing and obtaining the offset of at least one of the plurality of variables of the application, the module performs attaching to an address space used by the application during real-time operation to obtain a value for one or more the plurality of variables during the real-time operation of the application using the offset (e.g., Fig. 1 – Infrastructure of the on-line monitoring for efficient shared memory programming; Sec. 1 – Introduction, 5th Par. - … additional tools are necessary in order to transform the user-unreadable monitored data in a more understandable and easy-to-use form …; Sec. 3.4 – Projecting Back to the Source Code, 1st Par. - … the visualizer offers a "Data structure" window to reflect this mapping …. Shows all the shared variables occurring in a source code … provides

users with a global overview of accesses to the complete working set of the application ...)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Tao-2 into the Sridharan's system to further provide a compile listing stored on a computer-readable medium having an address map with an offset associated with each of a plurality of variable of an application; and a module stored on a computer-readable medium that performs reading of the compile listing and obtaining the offset of at least one of the plurality of variables of the application, the module performs attaching to an address space used by the application during real-time operation to obtain a value for one or more the plurality of variables during the real-time operation of the application using the offset in Sridharan system.

The motivation is that it would further enhance the Sridharan's system by taking, advancing and/or incorporating Tao-2's system which offers significant advantages that presents such a visualization tool displaying the monitored data in a user understandable way thereby showing the memory access behavior of shared memory applications as once suggested by Tao-2 (e.g., Abstract, Lines 8-15).

27. **As to claim 23** (Currently Amended), Tao-2 discloses the system wherein the module is further operable to search the compile listing and display the plurality of variables of the application for selection by a user (e.g., Fig. 1 – Infrastructure of the on-line monitoring for efficient shared memory programming; Sec. 1 – Introduction, 5th Par.

- ... additional tools are necessary in order to transform the user-unreadable monitored data in a more understandable and easy-to-use form ...; Sec. 3.4 – Projecting Back to the Source Code, 1st Par. - ... the visualizer offers a "Data structure" window to reflect this mapping .... Shows all the shared variables occurring in a source code ... provides users with a global overview of accesses to the complete working set of the application ...).

28. **As to claim 26** (Original), Sridharan discloses the system wherein the address space is further defined as a memory space and wherein the module attaches, using a socket layer, to the memory space used by the application (e.g., Abstract - a non-intrusive, reusable framework for collecting performance statistics of CORBA-based distributed systems).

29. **As to claim 28** (Currently Amended), Tao-2 discloses the system wherein the monitor is further operable, using the compile listing, to query the address map for one or more of the plurality of variables of the application (e.g., Fig. 1 – Infrastructure of the on-line monitoring for efficient shared memory programming; Sec. 1 – Introduction, 5th Par. - ... additional tools are necessary in order to transform the user-unreadable monitored data in a more understandable and easy-to-use form ...; Sec. 3.4 – Projecting Back to the Source Code, 1st Par. - ... the visualizer offers a "Data structure" window to reflect this mapping .... Shows all the shared variables occurring in a source

code ... provides users with a global overview of accesses to the complete working set of the application ...)

30. **As to claim 32** (Currently Amended), Tao-2 discloses the system further comprising a display component operably coupled to the module to perform receiving the value for the one or more of the plurality of variables, the display component operable to perform displaying the value (e.g., Fig. 1 – Infrastructure of the on-line monitoring for efficient shared memory programming; Sec. 1 – Introduction, 5th Par. - ... additional tools are necessary in order to transform the user-unreadable monitored data in a more understandable and easy-to-use form ...; Sec. 3.4 – Projecting Back to the Source Code, 1st Par. - ... the visualizer offers a "Data structure" window to reflect this mapping .... Shows all the shared variables occurring in a source code ... provides users with a global overview of accesses to the complete working set of the application ...)

31. Claims 22, 24-25, 27, and 29-31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sridharan in view of Tao-2 and further in view of Huang.

32. **As to claim 22** (Currently Amended), Tao-2 discloses the system wherein the module is further operable to read the compile listing and convert at least one of the plurality of variables (e.g., Fig. 1 – Infrastructure of the on-line monitoring for efficient shared memory programming; Sec. 1 – Introduction, 5th Par. - ... additional tools are

necessary in order to transform the user-unreadable monitored data in a more understandable and easy-to-use form …; Sec. 3.4 – Projecting Back to the Source Code, 1st Par. - … the visualizer offers a "Data structure" window to reflect this mapping …. Shows all the shared variables occurring in a source code … provides users with a global overview of accesses to the complete working set of the application …); and Huang discloses the associated offset (e.g., Sec. 3.2 – Typed Memory Objects – when an object is mapped, it can be read or written by simply reading or writing an address location within the address space corresponding to the offset of the byte in the object).

33. **As to claim 24** (Original), Huang discloses the system wherein the module is responsive to selection by the user of one of the plurality of variables to obtain the value for the selected one of the plurality of variables using the offset to locate the value of the variable in the address space (e.g., Sec. 3.2 – Typed Memory Objects – when an object is mapped, it can be read or written by simply reading or writing an address location within the address space corresponding to the offset of the byte in the object; Sec. 4.3 – Programming Interface, 1st Par. – *AcquireLock*()– acquires a lock for an object fragment specified by object, offset and length, *ReleaseLock*()).

34. **As to claim 25** (Original), Tao-2 discloses the system wherein the module is further operable to display the selected one of the plurality of variables (e.g., Fig. 1 – Infrastructure of the on-line monitoring for efficient shared memory programming; Sec. 1 – Introduction, 5th Par. - … additional tools are necessary in order to transform the user-

unreadable monitored data in a more understandable and easy-to-use form …; Sec. 3.4 – Projecting Back to the Source Code, 1st Par. - … the visualizer offers a "Data structure" window to reflect this mapping …. Shows all the shared variables occurring in a source code … provides users with a global overview of accesses to the complete working set of the application …)

35. **As to claim 27** (Original), Huang discloses the system wherein the module attaches, using the offset, to the memory space used by the application via an operating system service (e.g., Sec. 3.2 – Typed Memory Objects – when an object is mapped, it can be read or written by simply reading or writing an address location within the address space corresponding to the offset of the byte in the object; Sec. 4.3 – Programming Interface, 1st Par. – *AcquireLock*()– acquires a lock for an object fragment specified by object, offset and length, *ReleaseLock*()).

36. **As to claim 29** (Original), Huang discloses the system wherein the module is further defined as a subtask of the operating system (e.g., Sec. 3.2 – Typed Memory Objects – when an object is mapped, it can be read or written by simply reading or writing an address location within the address space corresponding to the offset of the byte in the object; Sec. 4.3 – Programming Interface, 1st Par. – *AcquireLock*() – acquires a lock for an object fragment specified by object, offset and length, *ReleaseLock*()).

37. **As to claim 30** (Currently Amended), Tao-2 discloses the system wherein the module is further operable to attach to the memory space where the application is operating and overwrite the value for one or more of the plurality of variables (e.g., Fig. 1 – Infrastructure of the on-line monitoring for efficient shared memory programming; Sec. 1 – Introduction, 5th Par. - … additional tools are necessary in order to transform the user-unreadable monitored data in a more understandable and easy-to-use form …; Sec. 3.4 – Projecting Back to the Source Code, 1st Par. - … the visualizer offers a "Data structure" window to reflect this mapping …. Shows all the shared variables occurring in a source code … provides users with a global overview of accesses to the complete working set of the application …); and

Huang discloses using the offset (e.g., Sec. 3.2 – Typed Memory Objects – when an object is mapped, it can be read or written by simply reading or writing an address location within the address space corresponding to the offset of the byte in the object; Sec. 4.3 – Programming Interface, 1st Par. – *AcquireLock*() – acquires a lock for an object fragment specified by object, offset and length, *ReleaseLock*()).

38. **As to claim 31** (Currently Amended), Tao-2 discloses the system wherein the module comprises: a reader component operable to perform reading the compile listing and further operable to perform converting at least one of the plurality of variables of the application to the associated offset; and a search component that performs receiving the associated offset of the at least one of the plurality of variables from the reader component, the search component operable to perform attaching to the application and

further operable to locate the value of the at least one of the plurality of variables (e.g., Fig. 1 – Infrastructure of the on-line monitoring for efficient shared memory programming; Sec. 1 – Introduction, 5th Par. - ... additional tools are necessary in order to transform the user-unreadable monitored data in a more understandable and easy-to-use form ...; Sec. 3.4 – Projecting Back to the Source Code, 1st Par. - ... the visualizer offers a "Data structure" window to reflect this mapping .... Shows all the shared variables occurring in a source code ... provides users with a global overview of accesses to the complete working set of the application ...); and

Huang discloses using the offset (e.g., Sec. 3.2 – Typed Memory Objects – when an object is mapped, it can be read or written by simply reading or writing an address location within the address space corresponding to the offset of the byte in the object; Sec. 4.3 – Programming Interface, 1st Par. – *AcquireLock*() – acquires a lock for an object fragment specified by object, offset and length, *ReleaseLock*()).

39. Claims 33-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sridharan in view of Tao-2 and further in view of Tao-1.

40. **As to claim 33** (Original), Tao-1 discloses the system wherein the display component is operable to employ the value to display a heartbeat (e.g., Sec. I – Motivation, 3rd Par., Lines 18-26 – it provides the programmer or system software with detailed information about all memory transactions over the network and the behavior of user-defined events).

41. **As to claim 34** (Original), Tao-1 discloses the system wherein the display component is operable to employ the value to display as a percentage complete heartbeat (e.g., Sec. I – Motivation, 3rd Par., Lines 18-26 – it provides the programmer or system software with detailed information about all memory transactions over the network and the behavior of user-defined events).

Claim Rejections – 35 USC § 102(b)

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102(b) that form the basis for the rejections under this section made in this office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

42. Claims 35-38 are rejected under 35 U.S.C. 102(b) as being anticipated by Kashima

43. **As to claim 35** (Currently Amended), Kashima discloses a system for non-intrusively monitoring COBOL application values, the system comprising: a memory area; a COBOL program stored on a computer-readable medium that generates program values and store the program values in the memory area during real-time operation of the COBOL program; and a COBOL monitor module stored on a computer-readable medium that shares the memory area with the COBOL program though a

technical layer (e.g., Fig. 1 – CORBA development environment using IDL, element of 'ORB libraries'; Sec. 2-2. CORBA Execution Environment, 1st Par. – At execution time, a function called ORB (Object Request Broker) plays a central role as a software bus for services such as server detection and request transfer. It makes it possible for clients to access distributed objects in the same way as through local access ...) and the COBOL monitor module reads the program values stored in the memory area by the COBOL program during real-time operation of the COBOL program (e.g., Sec. 1-4, 3rd Para. – in the case of CORBA, the connectivity with current system is kept high because of its mainframe and COBOL support); Sec. 2, 2nd Para. – the CORBA specification defines IDL, mapping to languages such as COBOL; Sec. 2-1, sub-sec. of 'Language Support' – the CORBA specification specifies the language mapping for COBOL; COBOL support is very important for the products that support mainframe systems).

44. **As to claim 36** (Original), Kashima discloses the system further comprising: a second COBOL program operable to generate second program values and store the program values in the memory area during real-time operation of the second COBOL program, and wherein the COBOL monitor module is further operable to read the second program values stored in the memory area by the second COBOL program (e.g., Sec. 1-4, 3rd Para. – in the case of CORBA, the connectivity with current system is kept high because of its mainframe and COBOL support); Sec. 2, 2nd Para. – the CORBA specification defines IDL, mapping to languages such as COBOL; Sec. 2-1, sub-sec. of 'Language Support' – the CORBA specification specifies the language

mapping for COBOL; COBOL support is very important for the products that support mainframe systems).

45. **As to claim 37** (Original), Kashima discloses the system further comprising: a second memory area; and a second COBOL program operable to generate second program values and store the program values in the second memory area during real-time operation of the second COBOL program, and wherein the COBOL monitor module is further operable to read the second program values stored in the second memory area by the second COBOL program (e.g., Sec. 1-4, 3rd Para. – in the case of CORBA, the connectivity with current system is kept high because of its mainframe and COBOL support); Sec. 2, 2nd Para. – the CORBA specification defines IDL, mapping to languages such as COBOL; Sec. 2-1, sub-sec. of 'Language Support' – the CORBA specification specifies the language mapping for COBOL; COBOL support is very important for the products that support mainframe systems).

46. **As to claim 38** (Original), Kashima discloses the system further comprising: a user interface operable to monitor and display the application values; and a client application in communication with the user interface and the COBOL monitor module, the client application operable to request the program variables of the COBOL program from the COBOL monitor module and provide the program variables to the user interface for display via the user interface responsive to a request from the user interface (e.g., Sec. 1-4, 3rd Para. – in the case of CORBA, the connectivity with current

system is kept high because of its mainframe and COBOL support; Sec. 2, 2nd Para. – the CORBA specification defines IDL, mapping to languages such as COBOL; Sec. 2-1, sub-sec. of 'Language Support' – the CORBA specification specifies the language mapping for COBOL; COBOL support is very important for the products that support mainframe systems).

Response to Arguments

47. Applicant's arguments filed on November 20, 2007 have been fully considered but they are not persuasive.

In the remarks, Applicant argues that, for examples:

a) None of the applied art teaches or suggests "a monitor that reads application values from a memory area used by the application during real-time operation (recited in REMARKS, page 14, 2nd paragraph, page 15, 2nd paragraph, page 16, page 17, 2nd paragraph)

Examiner's response:

a) Tao-1 (*Understanding the Behavior of Shared Memory Applications Using the SMiLE Monitoring Framework*) teaches "The visualizer and the adaptive run-time system described above are currently being developed with the SmiLE project. A mechanism for mapping each memory location observed by the monitoring system to its

corresponding program data structure identifier (procedure and variable names) is being implemented as well. This mechanism aims at giving the programmer a visualization of the program's memory behavior in familiar terms. Together with the monitoring data, the mapping mechanism helps the programmer optimize their programs so as to reduce the effect of remote memory accesses on performance ..." (see Sec. IV. – Designing A Tool Environment on Top of The SmiLE Monitoring Approach", 5th paragraph – emphasis added)

b) Note that examiner relies upon Tao-2, art made of record *(Visualizing the Memory Access Behavior of Shared Memory Applications on NUMA Architectures),* for teaching "reading plurality of variables of the application" (e.g., currently amended claim 21)

Conclusion

48. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Examiner, Art Unit 2192
April 9, 2008

/Tuan Q. Dam/

Supervisory Patent Examiner, Art Unit 2192